# Preserving Process: Hyperlink-Based Workflow Representation

*Explores the effectiveness of representing implementation-independent work and process flows as systems of hyperlinks that define the possible state transitions for managed objects. Identifies a set of functional requirements that a link-based workflow approach must satisfy and identifies alternative implementation approaches, especially in the context of generalized content and information management systems. Compares a declarative link-based approach to purely scripted forms of workflow definition. Analyzes the utility of generalizations provided by the XLink, XTM, and HyTime standards to the general problem of defining re-usable workflows that are largely implementation independent. Also explores the interactions of this approach to workflow management with versioning of the managed objects (that is, what, if any, implications are there for versioning and version representation on this form of workflow representation and management?). This exploration takes as its starting point the typical information management requirements of large enterprises (e.g., managing aircraft documentation, software documentation, legislation, etc.), where difficult problems of version management, traceability, repeatability, and configuration management cannot be avoided.*

*The general approach uses topics (in the topic map sense) to define the possible states in the workflow and links among the topics to define the possible transitions among the states (there the concept of "topic" is borrowed from topic maps for convenience--this paper does not explicitly explore the potential benefit of applying full topic map semantics to a workflow definition--that is left as possible further research). This system of topics and relationships thus defines an abstract workflow (or "workflow template"). Applying a workflow template to a real process in order to manage objects requires two things: an active management agent that moderates the movement of managed objects from state to state and a repository of managed objects to which the workflow states are bound. Managed objects are bound to a state by being made an occurrence of the state-defining topic. A managed object is made an occurrence when the workflow agent traverses a state-transition link from the managed object to the new state. The result of this traversal triggers an event which causes the target state-defining topic to be updated to add the new occurrence. When the managed objects are also managed as versioned objects, the transition to the new state may or may not be to a new version of the managed resource. In addition to simply binding objects to state-defining topics, this approach also explores how this transition can interact with or augment various forms of configuration management and the implications for managing sets of related objects as units of work.*

*The paper explores alternative ways of binding scripts and transition policies to states and transition arcs. Also explores issues of syntactic representation for archiving and interchange of work flows. Presents at least one prototype implementation of the system using a commercially-developed information management, link management system, with an analysis of the strengths and weaknesses of the approach. Concludes with a definition of the functional requirements that an information management system must satisfy in order to support this form of generalized workflow representation.*

W. Eliot Kimber, DataChannel, Inc.

## Introduction

The general subject of workflow is broad and somewhat confused, with no obvious consensus of even what workflow is, and certainly different definitions of workflow in different industries and

different use case contexts. For the purposes of this paper, I define workflow as "the management of information objects through a set of state transitions as requested by some active agent in the performance of a business process," where an "information object" is whatever the participating information management system manages, i.e., documents, product models, database records, etc, and an "active agent" is anything that interacts with the workflow management system to cause the state of objects to change (i.e., a human user or a software agent). The information objects may or may not be versioned. This definition focuses on the abstract aspects of workflow, leaving the details of how the abstraction might be used in a particular use case as a separate discussion. There are many useful ways to implement work flows and expose them to users of systems. The goal of this paper is to explore the essential abstractions of workflow and workflow management that underlie any particular workflow implementation.

Given such an abstraction, it should be possible implement generic workflow management services that can be quickly bound to use-case-specific user interfaces and business logic. Given a general abstract model for workflows, this paper then explores implementing those models using the hyperlinking structures of the XLink/HyTime and XTM/Topic Map standards in order to determine if the use of hyperlinks as an implementation approach adds compelling value.

Note: For the purposes of this analysis, XLink and HyTime are essentially identical and I use the two interchangeably. For the purposes of this discussion, the only important differences between the two standards are the syntactic options for representing hyperlinks in XML documents and the fact that HyTime has a formal notion of bounded object sets while XLink does not. Likewise, the XML Topic Map (XTM) and ISO/IEC Topic Map standards are functionally identical for the purposes of this paper and are, again, used interchangeably. An implementation of the ideas in this paper can choose to use either the W3C specifications or ISO standards.

This paper does not make direct reference to the workflow models defined by the WfMC (Workflow Management Coalition) . I found those models to be insufficiently abstract for my purposes here. However, given an appropriate abstraction, it should be possible to then implement the WfMC models as applications of the abstraction.

## Workflow Management Requirements

Workflow management appears to consist of two basic tasks: defining abstract workflows and applying those workflows to managed objects (concrete workflows). Workflows may be generic in that they may be applied to many use instances or they may be single-use, applying to a specific set of managed objects.

The overall workflow requirements addressed in this discussion are:

1. Must be able to define abstract work flows that are implementation independent. That is, abstract work flows express the details of general business processes in a generic way. For example, such an abstract workflow might be used to document processes governed by ISO 9000.
2. Must be able to generate concrete workflow instances from abstract work flows. That is, given an abstract work flow, generate a workflow instance that governs specific managed objects.
3. Workflow instances must be easily bindable to processors to make the workflows active.

## Why Hyperlinks?

This paper explores the hypothesis that hyperlinks are an appropriate and useful way to implement a workflow system integrated with more general content management systems. As explained in more detail in the next section, workflows can be modeled as transition graphs. If each state in a workflow is a link anchor, then clearly the links can represent the transition arcs between the nodes. That links can be used to model graphs doesn't necessarily mean that it actually helps to do so. Given a reasonable abstract model for work flows, an obvious approach is simply to implement the model in some appropriate programming language and there is no particular barrier to doing so with the abstract model defined in this paper. However, if there exists a system that provides generic hyperlink management facilities then there may be ways that such a system can be used to implement workflow in a way that is less costly and easier to maintain than a purpose-built system.

Using hyperlinks to represent workflows could provide the following benefits:

1. Implementation-independent workflow definition.

   By using hyperlinks to define workflows, the workflows are inherently implementation independent. The workflows are made active through "style sheets" that implement the semantics of the workflows. The style sheets may be system specific, but the business processes that the workflows define would not be.

2. Archival representation of workflow definitions and instances.

   The same characteristics that make link-based workflows system independent also make them an archival format.

3. Use of workflow definition documents as general-purpose process definition documents.

   Because the hyperlinks can be bound to other information in the same physical or logical documents, the workflow definition documents could serve double duty as, for example, ISO 9000 process documents.

4. Management system independence.

   Given two systems with equivalent link management features, it should be possible to bind the same workflow implementations and definitions to both systems at a reasonable cost.

5. Re-use of existing information management facilities to manage workflow definitions and instances.

   Because the workflow definition documents are just XML documents, they can themselves be managed using the same management facilities as any other documents. Because workflows are an important corporate resource, it seems advantageous to be able to manage them as you would anything else. Of course, this benefit would accrue to any declarative form of workflow definition.

Of course, the use of hyperlinks also adds complexity that may outweigh the benefits. This hypothesis also presupposes a fairly sophisticated link management infrastructure. As it happens, such an infrastructure exists, albeit in a form that is, unfortunately, currently only available to DataChannel and its customers (as engineers it would be our preference to make the system

we're building available as open source--however, business considerations have so far prevented that option).

This paper proceeds as follows: First it defines an abstract data model for representing workflows. This abstract data model could be implemented in any number of ways. It then defines an XML-based implementation that uses constructs from the HyTime and XLink standards to represent workflow definitions and instances. Finally, it describes how these documents could be combined with the generic management facilities provided by an generalized link management system to implement active workflows.

The next section describes how work flows can be viewed generically as finite state machines and thus transition graphs.

## Workflows Are Finite State Machines

If a workflow manages the transition of objects from state to state, it follows that a workflow is essentially defining a finite state machine. A state machine defines a set of states, the preconditions for entry to a particular state, and the possible next states. The state machine governs the transition of an object from state to state by matching the properties of the object or the overall system to the preconditions of the possible target states and selecting the state whose preconditions match. In the case of work flows, the states represent different stages in a process and the preconditions reflect either direct properties of the managed object (document type, owning group, classification, natural language, etc.) or general system properties (current time, states of other objects, etc.). In addition, in a workflow, a given state may define preconditions that must hold after an object has moved to that state (e.g., changing metadata on the object, generating new objects, setting global workflow properties, etc.).

The actions that cause a object to move from one state to the next are the "tasks" that the agents involved in the workflow perform. In some workflow systems the focus is on the tasks, rather than on the transitions between the states that the tasks cause. These systems focus on the mechanics of how the tasks are performed and often fail to fully express the reasons for why the tasks are being performed (the state changes) or even clearly define the states involved. However, in a more abstract workflow model, the details of how tasks are performed are much less important than defining what the result of having performed the tasks must be. This is analogous to defining interfaces in object-oriented programs: what's important is defining the requirements implementing objects must meet, not defining how they must meet them. Of course, in the implementation of a given workflow, it may make more sense to the users of the systems to put the focus on tasks (after all, their purpose is to perform the tasks), but that sort of redirection of focus is a user interface issue that should not affect the higher-level abstractions on which the work flow model is based. If a goal is to be able to re-use work flow definitions in different implementations, then it is essential that the details of how be clearly separated from the requirements of what (just as in XML the goal is to separate the core information content from the details of how that content might be presented or transformed for a particular use of that content).

Thus, while it is possible to model workflow as a series of actions to be performed rather than as a series of states to move through, it appears that the more appropriate abstraction for modeling work flows is as state machines.

State machines are often modeled as tables, where each row is a state and each column is a potential next state. For each starting state, the columns indicate which other states are allowed transitions. The table below shows a simple finite state table that defines the rules for constructing an XML start tag without attributes. It can be used either to parse existing strings or construct new strings. The task of constructing a new string is essentially a workflow where the precondition for each state is the addition of a character that matches an allowed state in the table.

| Current State | < | Name Character | " " | > | End |
|---|---|---|---|---|---|
| Start | x | | | | |
| "<" | | x | | | |
| Name Character | | x | x | x | |
| " " | | | x | x | |
| ">" | | | | | x |

**Figure 1. Simple XML Tag Construction Finite State Table**

To start the workflow, you create a managed object with a single "data" property that will contain the string constructed as a result of the workflow. To enter the workflow you start at the state labeled "Start". The Start state has no precondition beyond existence of the managed object. The "x" in the "<" column indicates that the only allowed transition is to the "<" state, meaning that you append the character "<" to the object. You make the data property value "<" and attempt to promote the object to the "<" state. The precondition is met, so the object moves to the "<" state. The table indicates that the only allowable transition requires appending an XML name character to the data. You append an "a" to the data, making it "<a" and attempt to promote the object to the Name Character state. The precondition that the newest character be a name character is met, so the object moves to the Name Character state. From this state there are three possible transitions: another name character, a blank, or a tag close (">"). You are happy with the tag as is, so you append ">" to the data and attempt to promote to the ">" state. The precondition is met so the object moves to the ">" state. From the ">" state the only possible transition is to the End state. You're done.

This form of state table is a convenient way to represent the implementation of a finite state machine but is not necessarily the best way to represent the transitions abstractly. A more effective abstract representation is often a transition graph, where each state is represented by a node and directed arcs between the nodes represent the allowed transitions. The state table above can be restated as this transition graph shown in Figure 2.

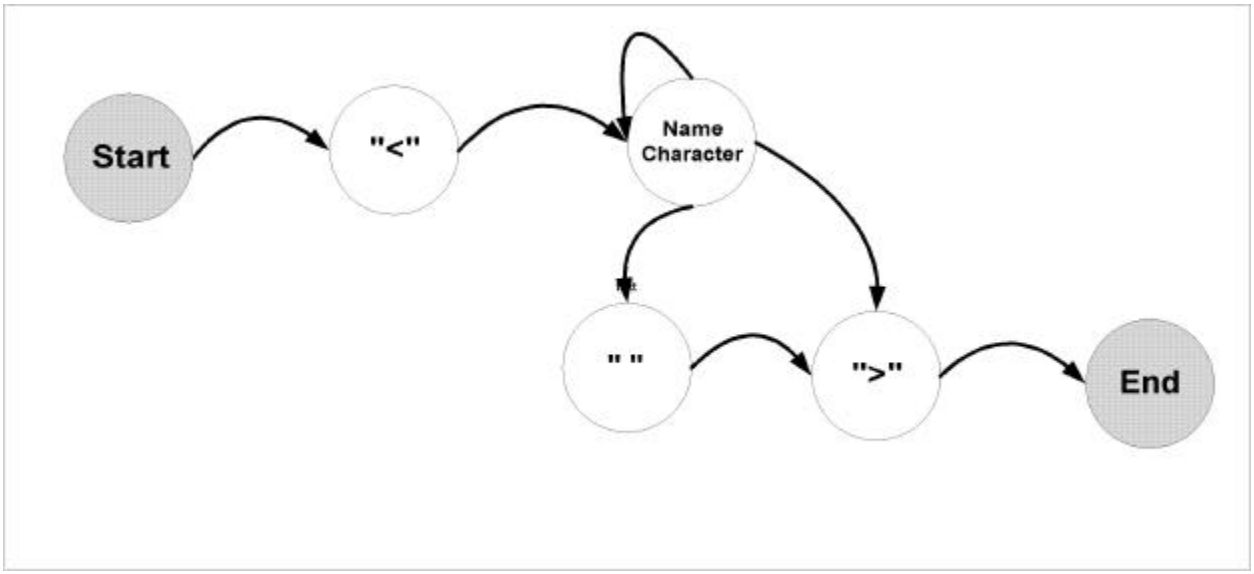


**Figure 2. Transition Graph for Simple XML Tag Construction**

The graph representation of the states and transitions makes it easier to see at a glance what the rules are. As a representation mechanism, it is easier to update the graph to reflect new rules simply by adding new nodes and updating the arcs.

Given that a workflow is a specialized form of state transition graph, it follows that workflows can be presented as graphs of states and arcs between the states. The agent tasks to be performed can be implicitly or explicitly defined by the arcs. Thus, any mechanism that can represent nodes and arcs between them is a candidate for the representation of work flows. XLink and HyTime-style hyperlinks explicitly represent relations among objects. Thus hyperlinks are an obvious candidate for representing abstract and concrete workflows where the links represent the transitions and the link ends represent the states.

# Workflow Abstract Analysis Model

As defined in the following data models, abstract work flows define a set of states and allowed transitions among those states without binding the states to any managed objects or participating active agents. Workflows become concrete when they are bound to specific objects, agents, and tasks the agents are to perform. The following UML models define a static data model for abstract and concrete workflows. This data model represents the highest (most abstract) level of analysis of what a workflow is, starting from the view that a workflow is essentially a transition graph. From this analysis level can be derived an implementation model that maps these abstract classes to hyperlinks and topics as defined in the XLink and HyTime standards.

NOTE: In this discussion, the term "abstract" can be used to qualify both workflows and the models in general. In this section the term "abstract" always qualifies "workflow" to distinguish abstract workflows from concrete workflows. All of the models in this section are abstract in the general sense in that they are not implementation models.

NOTE: Each of the UML diagrams in this section shows only part of the whole model. Diagrams showing the complete models are provided at the end of this section.

The UML models are organized into two packages: Abstract_Workflow and Concrete_Workflow.

# Abstract_Workflow Package

The Abstract_Workflow package defines the classes necessary to enable the definition of abstract workflows. The main requirements are the ability to define the individual states in the workflow, the allowed transitions from state to state, the preconditions that govern state transition, and the post conditions that must hold when an object has transitioned to the state.

As shown in Figure 3, an abstract work flow aggregates some number of work flow states. An abstract work flow state has zero or more preconditions, zero or more post conditions, and zero or more next states. By relating abstract work flow states to each other, the flow through the work flow can be defined.

An abstract workflow may have some set of properties. These properties may be referenced by preconditions or post conditions.

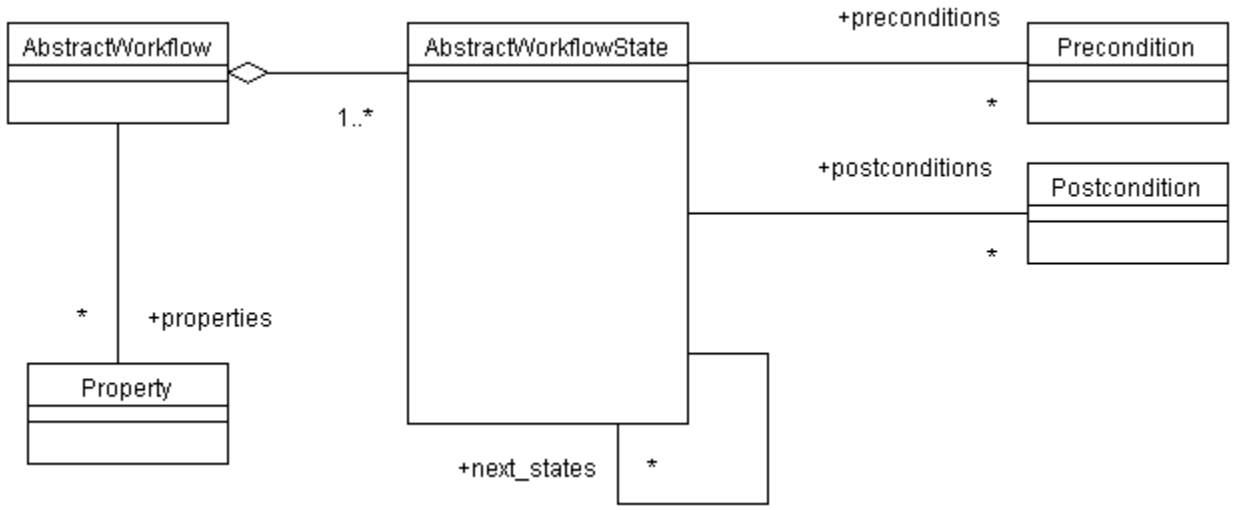


**Figure 3. Abstract Workflow**

Preconditions and post conditions are shown in Figure 4. Preconditions and post conditions both bind a single rule to some set of properties. A rule is an expression of some form (left undefined at this level of analysis) that evaluates the values of one or more properties and returns true or false. Precondition rules must return true before the state is entered and post condition rules must be true after the state is entered. Note that this leaves unstated how the properties that satisfy the post condition become true, as that is an implementation detail.

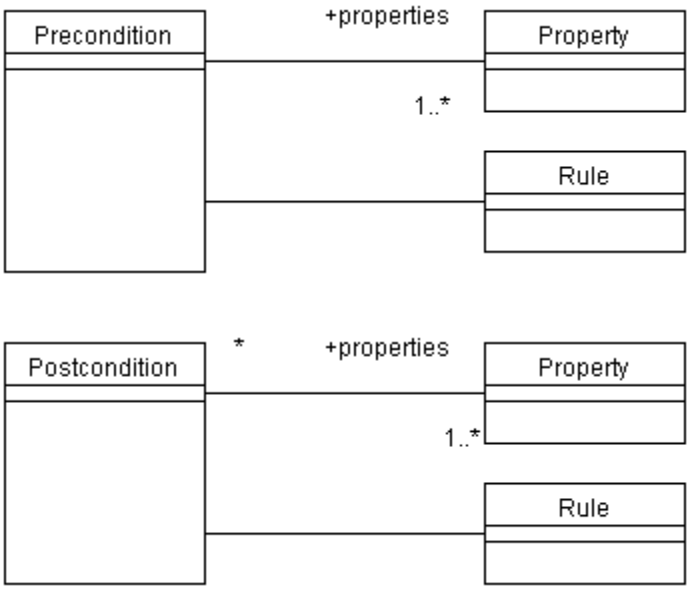


**Figure 4. Preconditions and Post conditions**

In addition to the workflow itself, there are two other classes that can have properties to which preconditions and post conditions may apply: information object classes and the information management system itself, as shown in Figure 5.

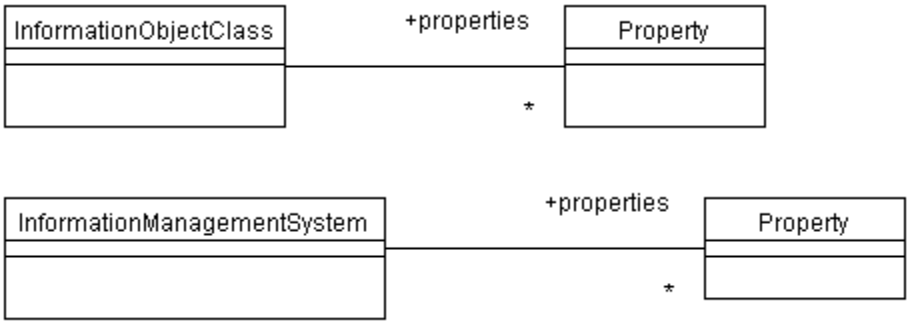


**Figure 5. Information Object and Information Management System**

Information object classes exist in this model simply to define properties that precondition and post condition rules may refer to. In the concrete model, information object classes are instantiated by information objects (naturally).

The information management system class is included here to again express the fact that the overall system may have properties to which preconditions and post conditions may refer (such as system time, local, authentication status, etc.). This model also introduces the concept of an information management system, which is the thing that uses the concrete work flows. But notice that at this level of analysis the only necessary connection between abstract workflows and the information management system is through the properties as used by preconditions and post conditions. That is because abstract workflows are simply defining the requirements for

instantiation of the workflows they define and are not necessarily bound to any particular system. Thus, the abstract workflow merely needs to say "in a concrete workflow instance of this workflow, the real management system must exhibit these properties".

That is the entire abstract workflow data model. At this level of analysis the relations between workflow states (the arcs in the transition graph) are represented by simple class associations in the UML, not by separate classes. It is not until we get to the implementation model for work flow that it becomes necessary to consider making the arcs classes.

## Concrete_Workflow Package

Concrete workflows serve to instantiate abstract work flows in order to bind them to specific information objects and the agents responsible for performing the work flow. The Concrete Workflow package defines the classes and actions that make up concrete work flows. Figure 6 shows the main classes that make up a concrete workflow.

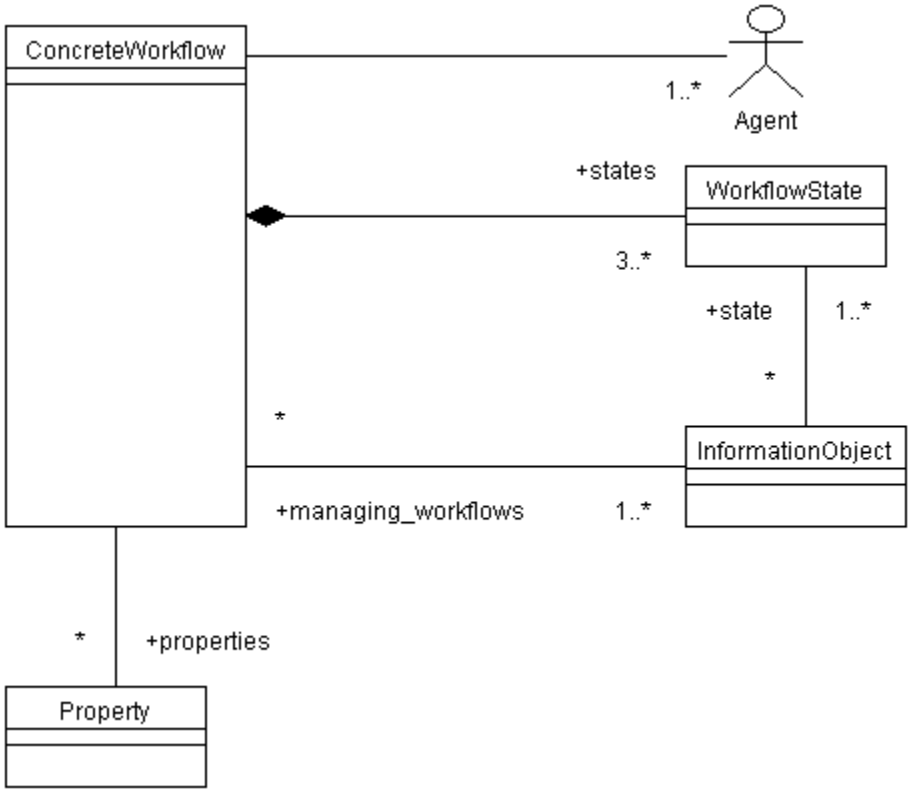


**Figure 6. Concrete Workflow Model**

A concrete workflow adds to the abstract workflow the information objects it is managing and the agents responsible for performing the workflow. Each information object has a single state within the context of a given workflow (but may be managed by more than one workflow at a time). Concrete workflows have properties corresponding to the properties of the abstract workflows they implement.

An agent is a human or software component that interacts with the information management system either to cause information objects to move from state to state within a workflow or to change the values of properties in order to satisfy the preconditions or post conditions of a particular state. For example, in an actual workflow system, a human user might request that an information object be promoted from one state to another. As the object enters the new state, the system might trigger a script that does something that causes the properties referenced by the post conditions to be set to the correct state (which might itself be the side effect of performing some other action, such as generating a rendition of a document or moving a object from one location to another). The concept of promotion is defined in Figure 7.

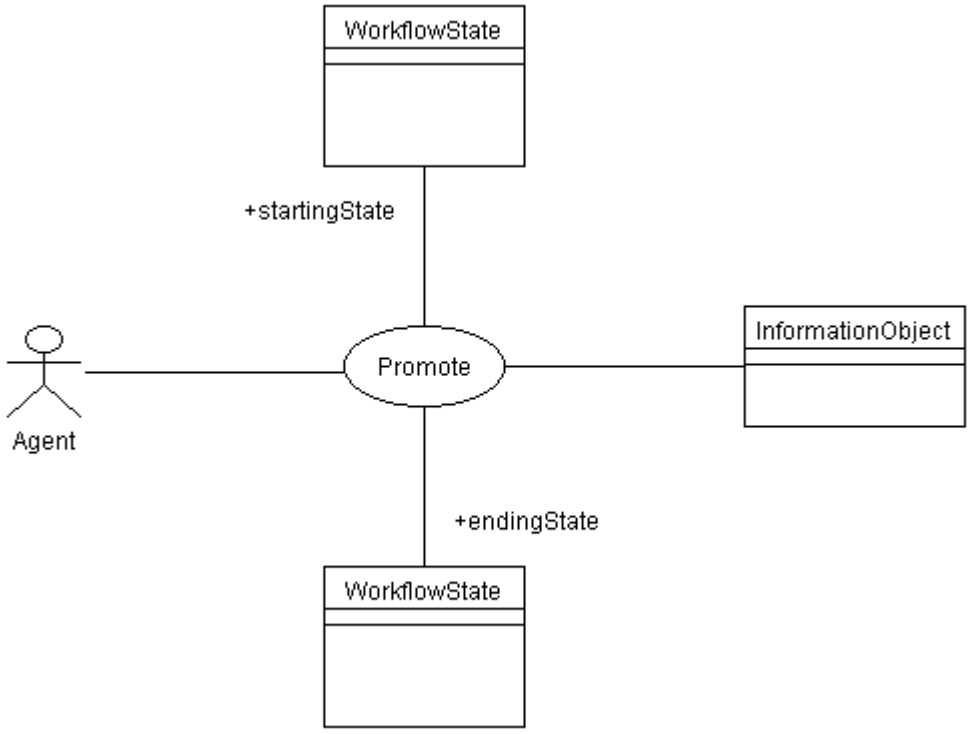


**Figure 7. Promotion**

The participants in this action are the agent, the starting workflow state, the information object, and the ending work flow state. The successful result of the Promote action is that the state attribute of the InformationObject now points to the endingState workflow state instead of the startingState workflow state.

Figure 8 shows the set property action.

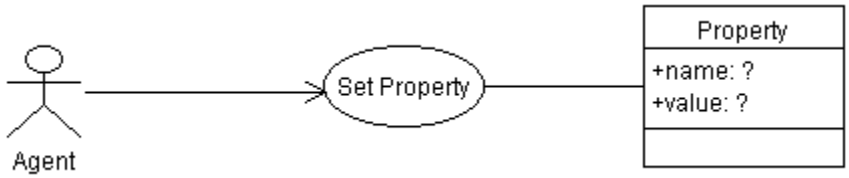


**Figure 8. Set Properties**

This diagram establishes that agents set properties. The result of a set property action is that the value of the property is different from its value before the set property action.

Figure 9 shows the WorkflowState class and its subclasses. Like AbstractWorkflowState, WorkflowState has preconditions and post conditions. For concrete workflows, there are three distinct subclasses of WorkflowState: StartState, IntermediateState, and EndState. The start and end states are defined in order to state the constraints that every workflow must have exactly one start state, one or more intermediate states, and one or more end states. An information object enters into a workflow by satisfying the preconditions and post conditions for the start state and exits a workflow by satisfying the preconditions and post conditions for one of the end states.

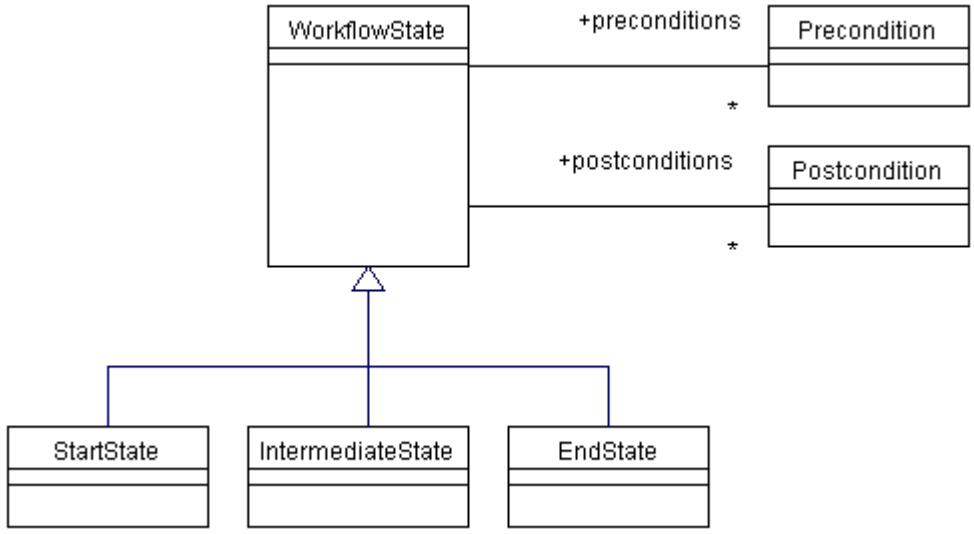


**Figure 9. Workflow State and Its Subclasses**

Figure 10 shows the concrete information management system. In the concrete model, the system explicitly manages all the information objects involved in workflows and has a direct relation to all the participating agents. The system also exhibits the properties required of it by the abstract workflows its concrete workflows are instantiating.

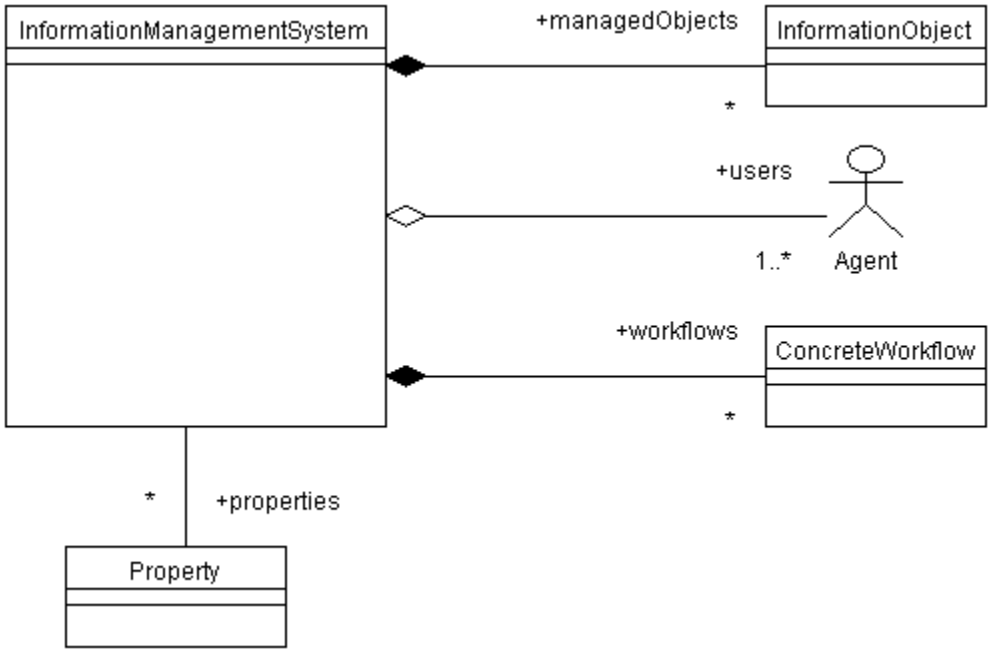


**Figure 10. Concrete Information Management System**

Figure 11 shows the instantiation relationships from the concrete work flow package to the abstract work flow package.

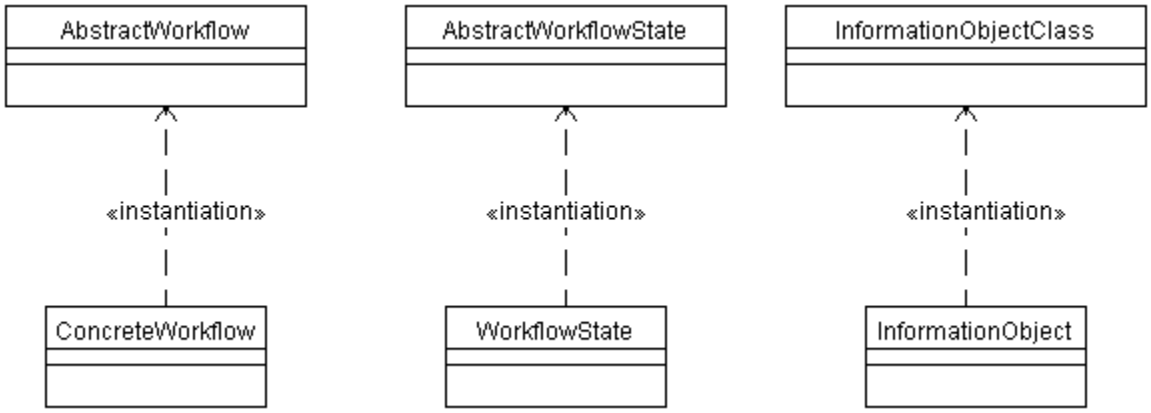


**Figure 11. Instantiation Relationships**

This diagram establishes the fact that a concrete workflow is an instance of an abstract workflow, a workflow state is an instance of an abstract workflow state, and an information object is an instance of an abstract information object class.

# Full Model Diagrams

The following figures provide the full model diagrams for the abstract and concrete workflows.

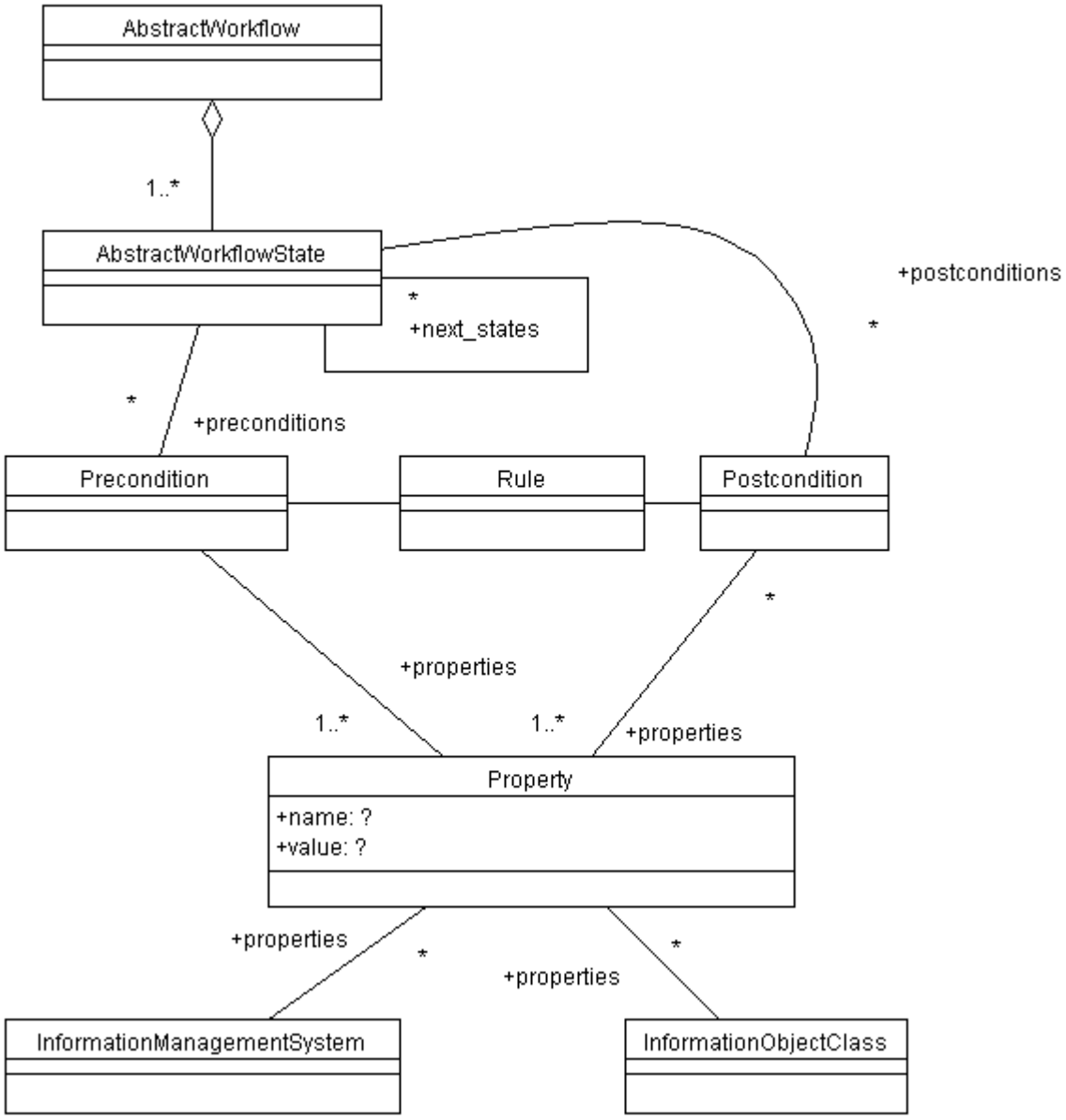


**Figure 12. Abstract Work Flow Full Model**

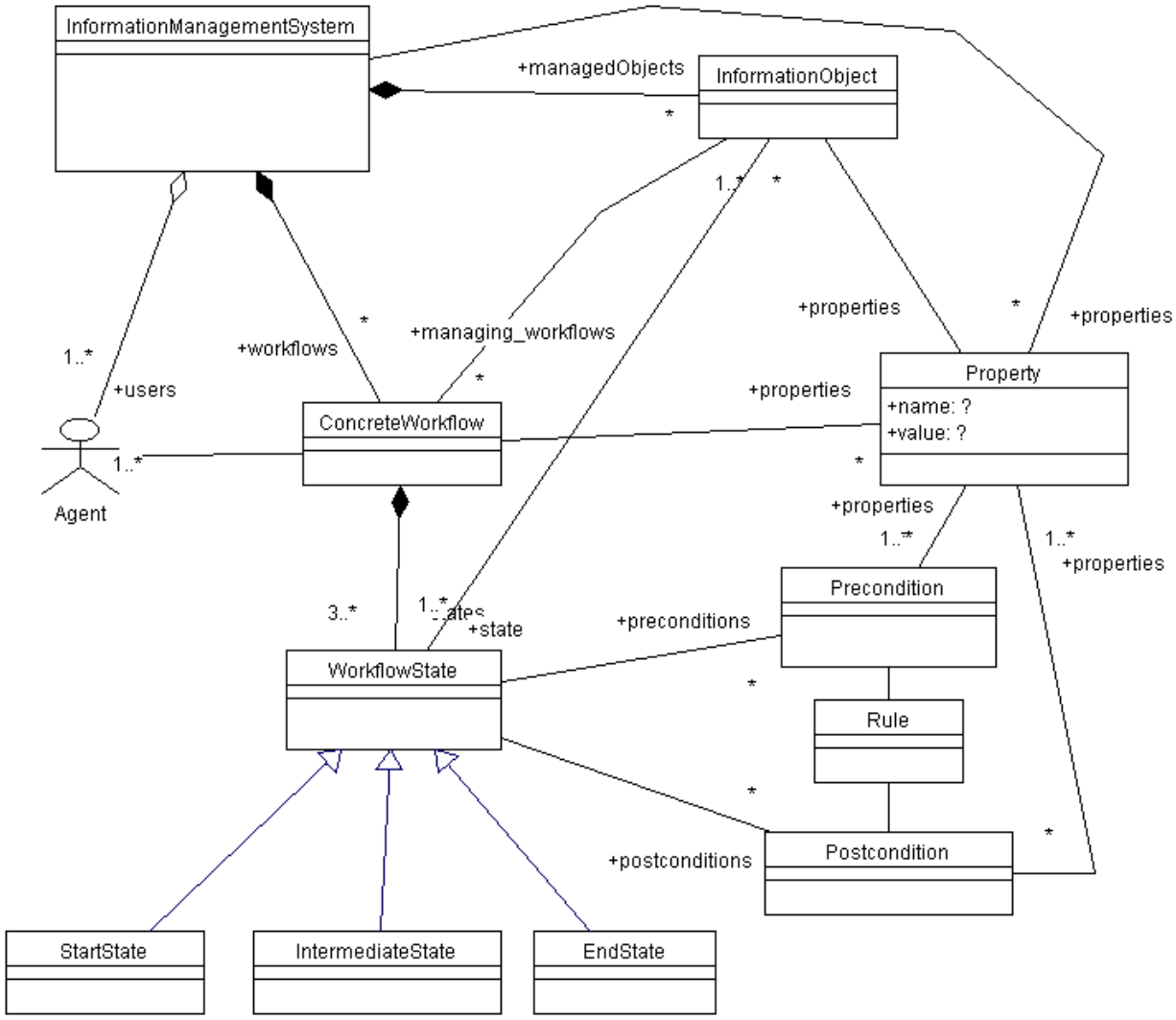


**Figure 13. Concrete Work Flow Full Model**

# Link-Based XML Representation Model

The purpose of the XML representation model is to enable the creation of documents that define workflow templates and workflow instances. Workflow template documents define abstract workflows that are then used to generate and govern concrete workflow instances. Concrete workflow documents represent workflow instances.

This design uses the XLink "extended" or HyTime "variable link" elements as the basis for the transition links.

Note: in my original formulation of the XML representation, I used topics and topic associations to represent states and transitions. However, I decided that the use of topics to represent states didn't add any value and could cause confusion with how relationships to managed objects are maintained syntactically. Without topics, topic associations add no value over and above generic

hyperlink elements. In any case, it was never my intent that a link-based workflow would be a full topic map.

Note that if concrete workflows are managed as literal XML documents (that is, as character strings conforming to the XML specification), then every time a managed object moves to a new state, a new version of the workflow document must be produced. If these versions are managed in a version management system, it automatically creates a record of the actions performed through the workflow. If the concrete workflows are managed as "in-memory" documents (e.g., DOMs that are dynamically mutated as managed objects move to new states), then there is no necessary saving of historical state, but no barrier to doing it either.

The XML design is organized into two distinct packages of element types that both import element types from a package of common element types. The Workflow_Template_DTD package defines the element types needed to define abstract workflows. The Concrete_Workflow_DTD package defines the elements needed to define concrete workflow documents. The Common_Workflow_Elements_DTD package defines element types that have identical semantics and constraints in both packages. Formally, these DTD-defining packages are implementation refinements of the abstract and concrete workflow data models.

Note that the DTD packages do not impose any particular storage organization constraints on the workflows--it would be possible to have a single XML document that contained both abstract and concrete workflows. However, normal data management practice would make it most convenient to separate abstract and concrete workflows into separate physical documents. It might also make sense to make each state in a concrete workflow into a separate document so that each state can be individually versioned without affecting any other states. In any case, the storage organization of the elements does not have any semantic significance.

When using the HyTime standard for hyperlinking semantics, a concrete workflow forms the hub of a hyperdocument that has as its BOS members the concrete workflow document, the governing abstract workflow document, an all managed objects (assuming that management is done at the entity level). When using XLink, one could define a link group that served to enumerate the documents participating in the workflow.

These DTDs can be used as the local DTDs for documents, as SGML architectures, or as name spaces. If the DTDs are used as the local DTDs for documents, then concrete workflows and workflow templates must be in separate documents. If architectures or name spaces are used then concrete and abstract workflows can coexist in the same document. The DTDs as defined here do not define any elements for holding documentation or implementation-specific metadata. Such extensions can be provided by creating derived DTDs or architectures that use these DTDs as their base architectures or by using additional hyperlinks to annotate the workflow documents.

## The Abstract Hyperlink Data Model

The XLink and HyTime standards both reflect the same high-level data model of what a hyperlink is. This model is explicit in the HyTime standard and implicit in the XLink standard. The key aspect of this model is the distinction between hyperlink anchors and the members of those anchors. In casual discussion of hyperlinks we usually use the term "anchor" to mean both the anchor objects and the anchor members without making a clear distinction. However, for the purposes of talking about the details of activity policies and traversal it is necessary to make a clear distinction between anchors and anchor members. In particular, when you do this it

becomes clear how a single object can be a member of more than one anchor. If this distinction is not made clearly, then the details of link processing can get very confused.

Figure 14 shows the basic hyperlink data model.

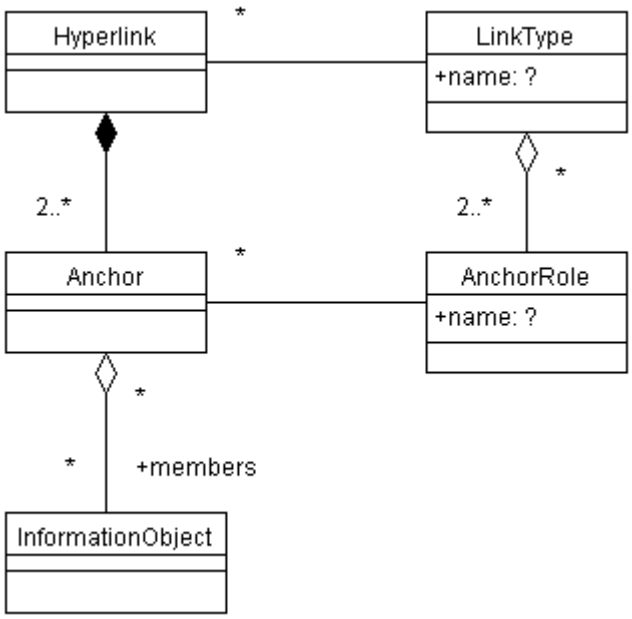


**Figure 14. Hyperlink Abstract Data Model**

A hyperlink has two or more anchors[1] Each anchor aggregates zero or more information objects. A hyperlink has a governing link type, which characterizes the link as a whole (equivalent to "role" on the "extended" element in XLink). Each anchor has an anchor role that is unique within the link type (that is, each anchor of a link must have a unique anchor role).

The purpose of the Anchor class is to characterizes its members with a particular role within the link. Traversal through a link is from anchor members to other members of other anchors of the same link.

If a given information object is a member of anchors of different links, there is no defined semantic implication of that fact--the object is simply participating in multiple links. However, for traversal purposes, if an information object is a member of two links of different anchors, then traversal to the object through one link constitutes external arrival at the object with respect to the other link. There is no defined notion of automatically chaining from link to link (that is, an implementing system would not, by default, automatically redirect through the information object from one link to another).

## Workflow_Template_DTD Package

The root of a workflow template is the WorkflowTemplate element type, as shown in Figure 15.

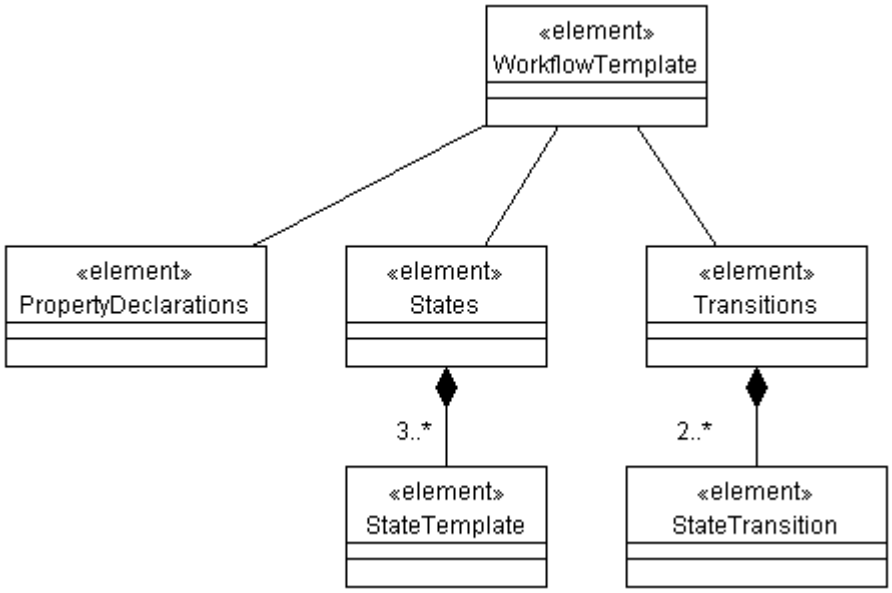


**Figure 15. Workflow Template Element Type**

A WorkFlow template contains a PropertyDeclarations element, which contains definitions of any properties used by preconditions or post conditions, a States element, which contains template definitions of all the states in the workflow, and a Transitions element, which contains the state-to-state transitions for the workflow. There must be at least three states (one start state, one intermediate state, and one end state) and therefore at least two state transition links.

Figure 16 shows the StateTemplate element type. A StateTemplate defines the preconditions and post conditions for the state. The Preconditions and post conditions elements are simply containers for Precondition and post condition elements, respectively. Precondition and post condition elements contain expressions in some expression language.

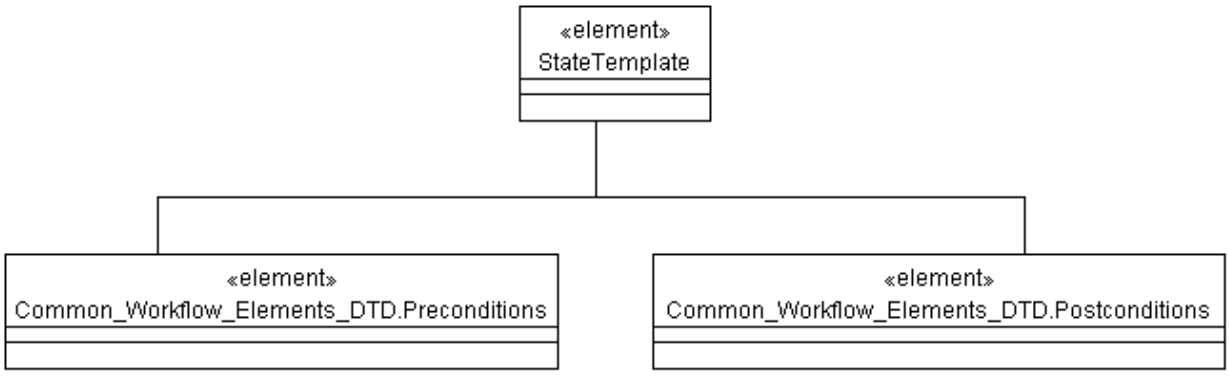


**Figure 16. State Template**

Figure 17 shows the PropertyDeclarations element type.

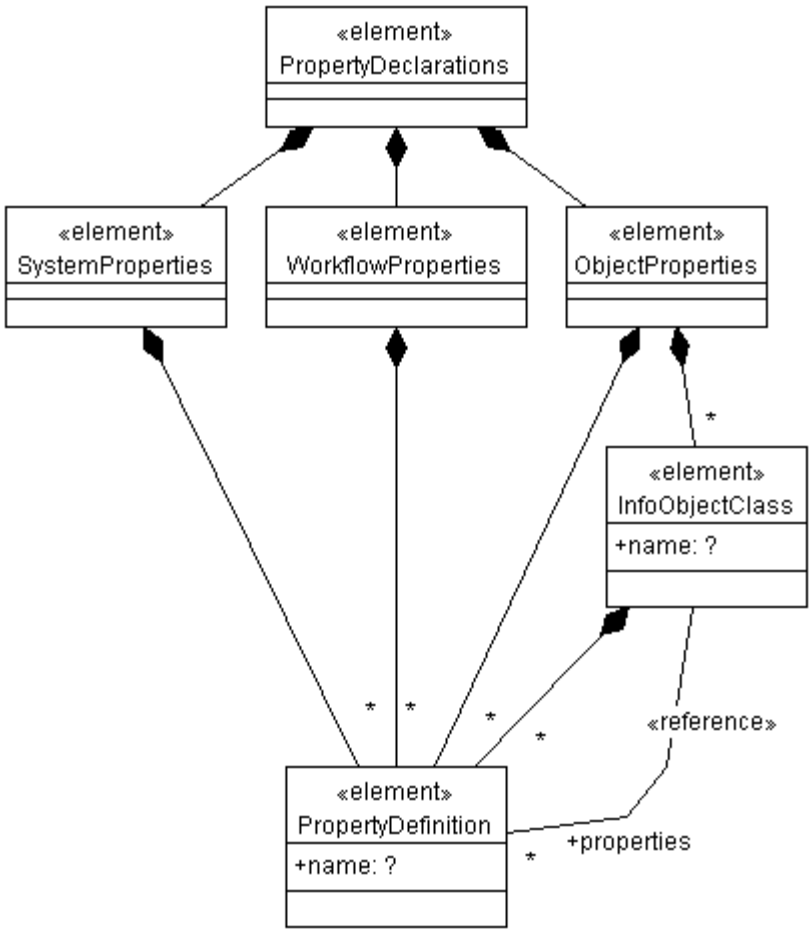


**Figure 17. Property Declarations**

The property declarations organize property definitions into system, workflow, and object properties. Each PropertyDefinition has a name attribute. The property definition names must be unique within the scope of their containers. Object properties that are not scoped to a particular InfoObjectClass may be used by any InfoObjectClass. An InfoObjectClass element can point to properties by name or directly contain properties unique to that object class.

The InfoObjectClass element is not intended to be a full class definition--it serves only to establish the names of the properties that a given object of that class may exhibit. The actual class definition will be a function of the larger information management system (e.g., an XML DTD, a business object model, etc.).

Figure 19 shows the formal refinements from the abstract workflow data model to the XML DTD implementation model. This diagram serves to establish the formal binding across these two levels of abstraction.

Figure 18 shows the StateTransition hyperlink. These hyperlinks relate one state to another state in order to define the allowed state transitions.

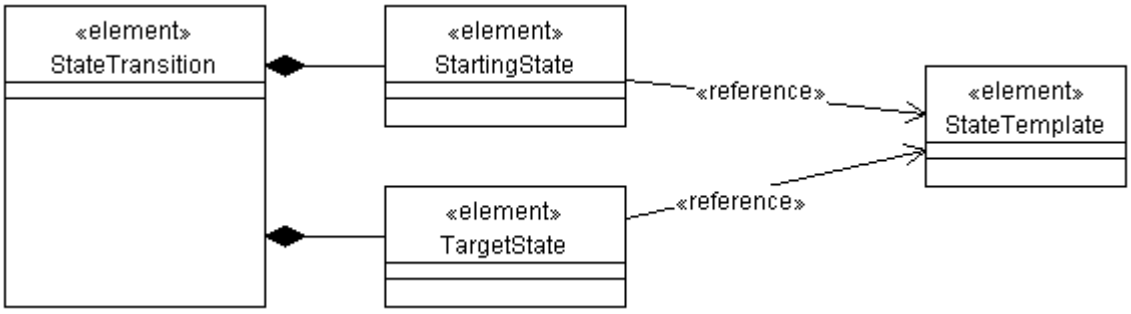


**Figure 18. StateTransition Element Type**

The StartingState and TargetState elements are then anchors of the StateTransition link. They point to StateTemplate elements.

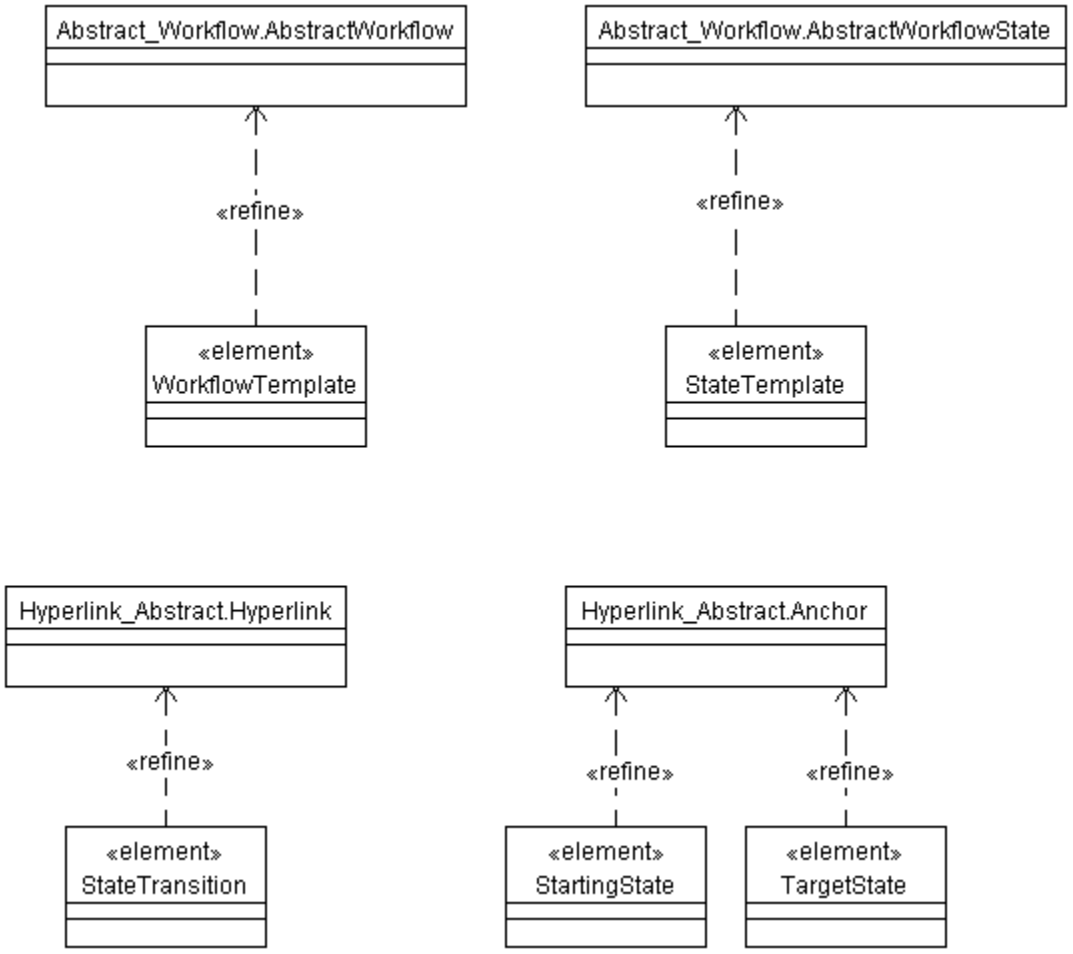


**Figure 19. Workflow Template Refinements**

## Concrete_Workflow_DTD Package

The concrete workflow DTD defines the elements needed to create workflow instances. Each concrete workflow points to the workflow template that governs it. Figure 20 shows the Workflow element type, which is the root of a concrete workflow (and normally, but not necessarily, the root of the XML document that contains the Workflow).

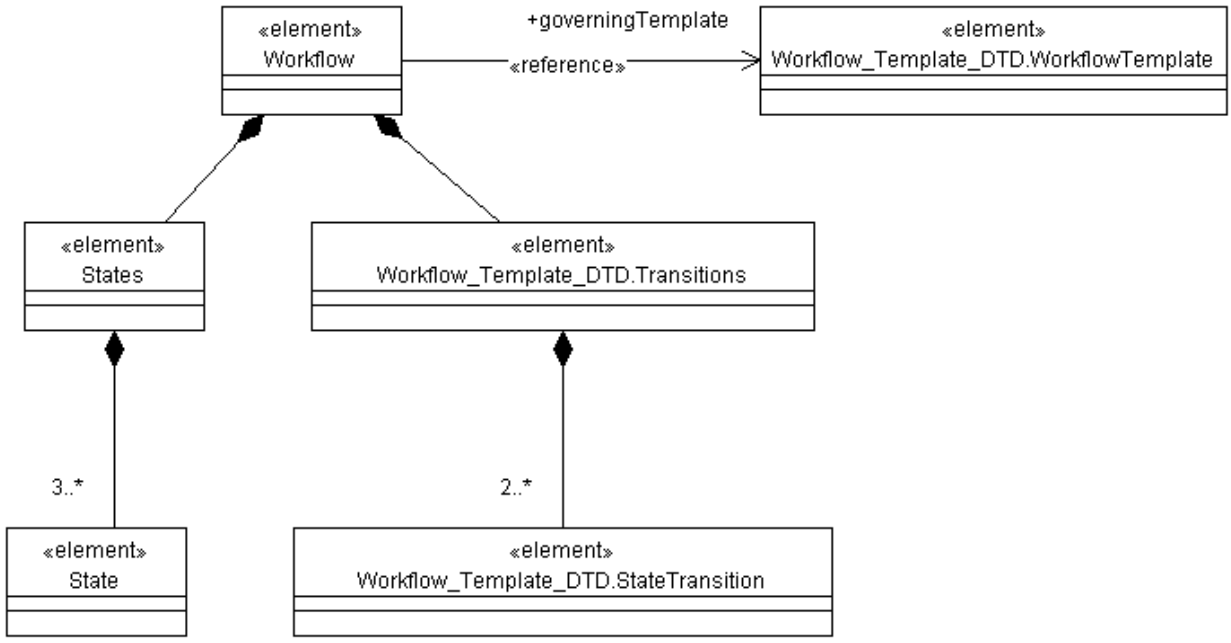


**Figure 20. Concrete Workflow Element Type**

A concrete workflow consists of three or more states and two or more transition links. The Workflow element has a direct reference to its governing template, which would normally be in another document (but could be in the same document).

Note that all the preconditions and properties are maintained only on the WorkFlow template. Pre- and post-condition expressions would normally be defined in terms of object classes, rather than specific object instances. However, a workflow that is only applicable to a specific set of objects can be defined by creating a workflow template whose preconditions refer to specific object instances rather than classes.

Figure 21 shows the concrete State element type. The concrete state differs from the abstract state in that it is bound to zero or more ManagedObject elements, which act as proxies for the actual information objects that are being managed. The relationship between the ManagedObject elements and real information objects is not shown in this diagram because there is no single definition of what an information object is outside of a HyTime context. In HyTime, an information object is always represented by a node in a grove or as an entity. In XLink, the notion of addressable resource is not well defined except for XML resources.

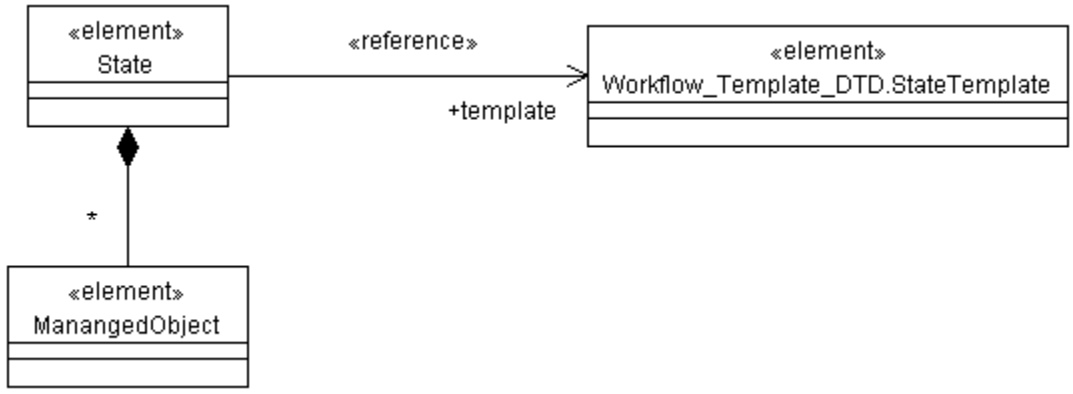


**Figure 21. State Element Type**

Each concrete state has an explicit pointer to the StateTemplate element of which the concrete state is an instantiation. The referenced StateTemplate must be in the workflow template referenced by the Workflow element of which the concrete state is a member.

Figure 22 shows the TransitionLink element. Transition links relate ManagedObject elements to their potential target states. Note that this is different from the abstract workflows, where the transition links relate states to state. Transition links also point to the starting state the information objects are in, but not as an anchor. In an XLink implementation this relation is primarily for information and convenience. In a HyTime implementation this relation can be used as the location source for the addresses of the contained ManagedObject elements.

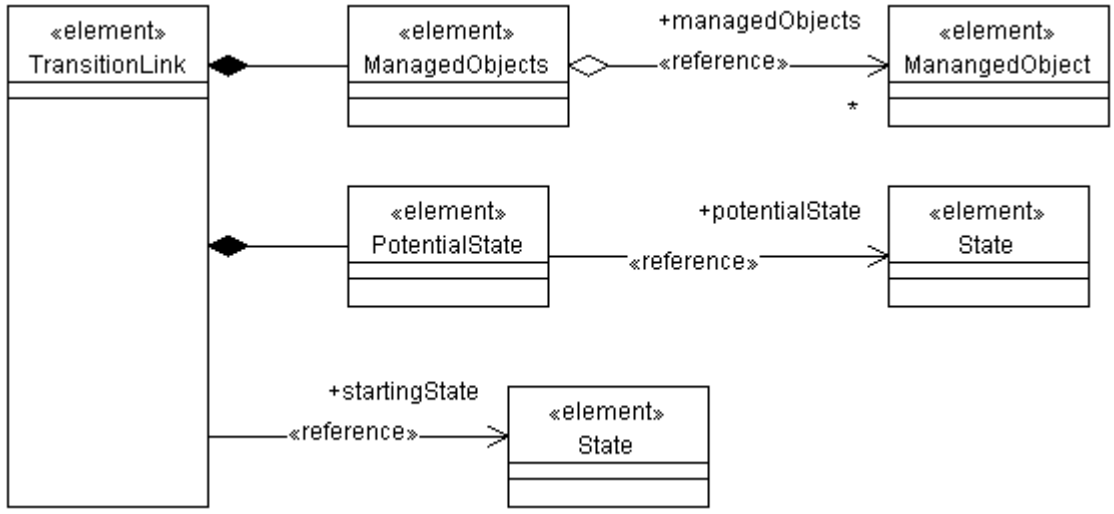


**Figure 22. Transition Link Element Type**

Figure 23 shows the refinements for the concrete workflow document type.

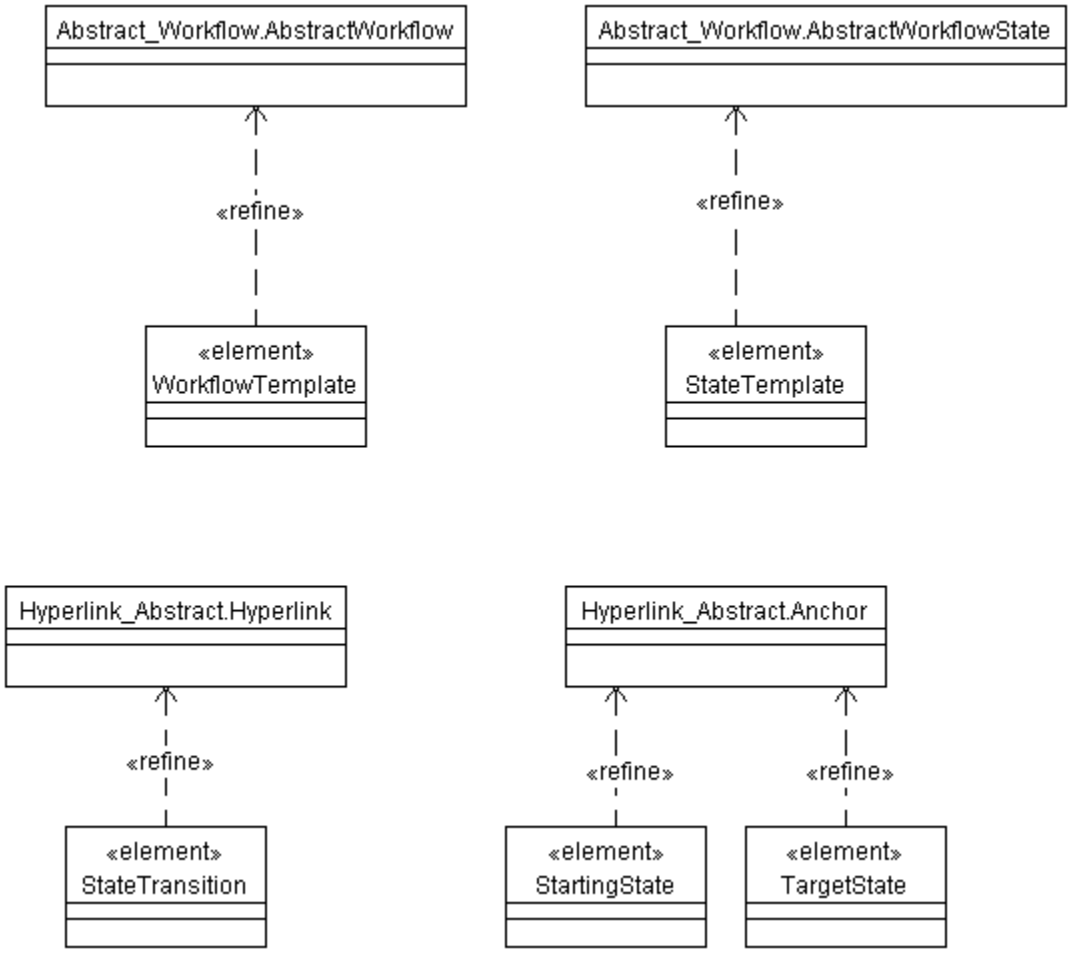


**Figure 23. Concrete Workflow DTD Refinements**

# Generating Concrete Workflow Documents From Workflow Template Documents

The purpose of workflow template documents is to define abstract workflows and then use those workflows as templates for generating concrete instances of the workflows. The generation process can be defined as follows (here making a concrete workflow a separate single document):

1. Create a new document whose root element is Workflow. Set the governingTemplate pointer to point to the input workflow template element. Create the required container elements within WorkFlow.
2. Declare as external data (unparsed) entities the entities that are or that contain the information objects to be managed.
3. For each StateTemplate in the workflow template, create a corresponding State element whose template attribute points to the corresponding StateTemplate element.

4. For the State that is the starting state for the workflow, create one ManagedObject element for each information object to be managed and set its pointer to the information object.
5. For each StateTransition in the workflow template, create a corresponding TransitionLink element whose PotentialState element points to the State element corresponding to the StateTemplate that is the potentialState anchor of the input StateTransition, whose startingState pointer points to the State element corresponding to the StateTemplate that is the startingState anchor of the input StateTransition, and whose ManagedObjects elements point to either nothing (if the potential state is not the start state of the workflow) or to one of the ManagedObject elements in the starting state.

The resulting concrete workflow document represents the initial state of the workflow. As managed objects are progressed through the states in the workflow or organized into units of work (by pointing to multiple ManagedObject elements from a single ManagedObjects anchor of a TransitionLink), new versions of the concrete workflow document are created.

# Making Workflow Documents Active

The workflow documents are, of course, static: they provide a declarative definition what the workflow is or a static record of the state of a concrete workflow is at any given moment in time. However, the XML alone is not sufficient to make the workflow active. The use of hyperlinks implies that a generic hyperdocument viewer would enable navigation among the workflow states and from states to managed objects, but wouldn't do anything to actually change state or implement checking of preconditions or post conditions.

Making the workflow active requires binding the following generic system features to the workflow documents:

- Interception of the "promote" action, which requires checking preconditions and post conditions for candidate target states
- Creation of new versions of concrete workflow documents
- Provision of some way for agents to preform the tasks necessary to satisfy preconditions and post conditions

In addition, an implementing system may want to reflect state changes in system-specific ways, such as moving objects into different physical or logical locations within a content management system.

## State Transition is Link Traversal

For a given concrete workflow, a given managed object will be a member of exactly one state topic and will have one or more transition links linking it to potential target states (unless the state is an end state). Promotion of the information object to a new state can be bound to the act of traversing the link.

For a hyperlink anchor, there are four possible actions related to the traversal of the link: arrival at an anchor, traversal from the anchor to any of the other anchors, traversal to the anchor, or departure from the anchor (having previously traversed to it).

Arrival means "the agent involved has access to the anchor and some or all of its members such that, were traversal allowed, the agent could initiate it by some means". Departure means "moving from the anchor to some other place by means other than traversal across the same anchor", e.g., scrolling away from the anchor. The arrival and departure actions are not directly useful for making workflows active but are necessary in order to completely define the distinction between movement among anchors of a single link and movement among anchors of different links.

Another key concept is the idea of "activity policies", which define what actions a given agent is allowed to perform on a given node (in the DOM or grove sense). Activity policies are associated to nodes in order to say "this action, when applied to this node, is governed by this policy". Normally policies are programs that determine whether the action is allowed and, optionally, perform other tasks that may have side effects (e.g., requesting a credit card number, logging the action, setting system state properties, etc.).

Given that hyperlink anchor members are objects to which activity policies can be applied and given that we have defined the four traversal actions of "arrival", "traverse from", "traverse to", and "depart", it follows that we can bind policies to these actions.

If the mechanism by which promotion is requested is the traversal of object-to-state links, then the different traversal-related actions can provide the triggers necessary to drive the workflow and the activity policies can implement the workflow actions.

## Activity Policies Implement Workflow Actions

To promote a managed object to a new state in a workflow, you must first arrive at the object (in its role as a member of a transition link anchor). This would normally be accomplished through some user interface that provides access to the workflows and enables traversal. This could be a completely generic hyperdocument browser that supports traversal activity policies or it could be a purpose-built workflow user interface. The net effect should be the same.

The user wishing to promote an object selects the object and initiates traversal. If there is more than one potential target, the system must present a choice of target anchors and the user must select one. In the case of workflow documents, transition links are always binary, so there will be exactly one traversal target (potential state) per transition link.

Having selected a target anchor member, the user requests the traversal, which triggers the check for any traverse-from activity policies. For workflow documents, there must be activity policies associated with the traverse-from actions that check the preconditions of target state to see if they hold and also check to see if the user has the rights to perform this action at all (irrespective of any work-flow-defined preconditions). If the preconditions do hold, then the traverse-from action is allowed to continue. If the preconditions do not hold, then the action is vetoed and traversal is blocked.

If traversal is allowed, then the traverse-to policy for the target anchor is triggered. The traverse-to activity policy can do anything but it must check that the post conditions are true. If the post conditions are true, the system must create a new version of the concrete workflow document that deletes the promoted information object from the traverse-from anchor and adds it to the traverse-to anchor. The traverse-to action is then allowed and traversal completes. Note that the completed traversal is to the version of the anchor as it existed before the traversal. Once the

traversal completes, the hyperdocument would be regenerated (or updated in place) to reflect the new version of the workflow document.

Note how this use of activity policies on traversals serves to separate the implementation of the workflow from the workflow definition documents and make clear where the integration point is, such that the activity policies could be adapted or migrated to other hyperdocument management systems that support traversal activity policies.

In addition to the traversal activity policies, other activity policies could further constrain an agent's ability to view or interact with the workflow. For example, a single workflow might manage objects for which not all users have view access, with "view" activity policies enforcing the view rights.

## Agent Tasks Set Properties

To promote an information object to a new stage normally requires that the object have been modified or versioned or that other actions have been performed (e.g., review, approval, etc.). How these tasks get performed is entirely implementation dependent but the end result of performing a particular task is that relevant properties of the information object, the workflow, or the information management system are set to new values. For example, a precondition might be that a validation process has been performed or that a specific user or set of users has indicated approval of the information object.

Thus, when enabling a set of workflow templates in a particular information management system, there is always the unavoidable integration task of binding specific agent actions to the setting of properties. However, because the traversal activity policies are all accessing the same general pool of properties, it seems reasonable to presume a general framework for setting and accessing workflow-related properties, which provides a fairly well-defined integration point for integrating tools that support tasks with the setting of properties that reflect the performance of those tasks.

## Potential Problems With This Approach

The use of activity policies implies some amount of administrative overhead to define user's rights with respect to particular action/object pairs. The complexity of this overhead is largely determined by the precision of control required by a specific use case, but in general it should be possible to take advantage of things like link types, element types, and user roles to define bindings more or less generically without the need to directly define policies for each anchor of each transition link.

# Implementation Scenario

This section describes a possible implementation scenario and simple use case based on the functionality of the Bonnell link and information management system being developed by DataChannel, Inc.

## Overview of the Scenario

In this scenario, a simple workflow template has been designed that models a basic "draft-review-publish" workflow. This workflow is then applied to two documents that are being developed as part of the same overall project (e.g., two documents in a product's documentation library). The scenario involves the following human agents:

- Analyst Arthur defines the workflow template document
- Manager Barbara creates a workflow instance and binds it to the two documents
- Author Carl is responsible for Book A
- Author Dawn is responsible for Book B
- Engineers Edgar and Francis are responsible for reviewing and approving both books

The scenario involves the following software agents:

- The workflow creation interface manages the task of creating concrete workflows from workflow templates and setting the initial system state.
- The workflow notification service sends notices to human agents when the state of the workflow changes in a way that requires them to act.
- The publishing server automatically generates publishable forms of documents for delivery in various forms when a document enters the "published" state.
- The workflow management user interface that lets users see and modify the state of the workflow (i.e., to promote managed objects to new states or set properties in order to satisfy preconditions or post conditions).

In this use case, Arthur defines the workflow template and adds it to the workflow system, adding the appropriate activity policies for each possible state transition and setting up any automated processes that serve to satisfy post conditions. Barbara creates a new workflow instance, binding the workflow to the documents Book A and Book B and assigning the other people to the appropriate roles in the work flow (author and reviewer).

## The Users' View of the Workflow

The following describes how the workflow appears to the human agents performing it:

1. The group that Barbara works for requests a new workflow to managed the development of user manuals. They contract with Arthur to define and implement the workflow, which will be called "develop user manual".
2. Arthur starts the process by analyzing the document development business process and capturing it as a workflow template document. He then determines how this workflow should be implemented at the current time in terms of the specific user tasks to be performed, the tools used to perform them (editors, reviewing tools, browsers, etc.) and the automated tasks to be performed, such as production of deliverable forms of the documents managed in the workflow. These implementation details are distinct from the structure of the abstract workflow and will change as the details of the business process and supporting software change. Arthur defines activity policy scripts and registers them

with the hyperlinking system. Because Arthur is working with a workflow template, the policies are associated with managed objects in terms of the links and anchor roles defined in the workflow template. Once he has the workflow template set up, he informs the group that requested the workflow that the "develop user manual" workflow is now ready for use.

3. Barbara accepts a new project that requires updating two documents, Book A and Book B. The two books are part of the same library of documents and must be developed in parallel. Using the Workflow creation user interface, Barbara selects the "develop user manual" workflow template. The system asks her to select the documents to be managed in this workflow. She uses the management system interface to locate the current versions of both documents and adds them to the workflow. The work that Arthur did including defining the user roles "author" and "reviewer", which are in turn bound to the mechanisms for setting various properties. For example, the author role must set the "validation complete" property to true, certifying that they have validated the document. This is a precondition for entry into the "under review" state. Reviewers have to set the "approved" property to true or false and all reviewers must indicate approval before the document can be promoted to the "published" state.

   Barbara sets some deadlines for entry into various tasks, which act as milestones (e.g., "approved" state must be entered no later than 5 Jun 2001).

   Barbara indicates that Carl is the author of Book A and that Dawn is the author of Book B. Likewise, she indicates that Edgar and Francis are reviewers of both documents. She saves the workflow and activates it by promoting both Book A and Book B to the "under revision" state. The system composes"you've got work" e-mails to Carl and Dawn and asks Barbara to refine and send both messages. She adds details about the assignments to the notes and fires them off.

4. Carl gets the "you've got work" email telling him he's got a new assignment. He opens the Workflow Management user interface and sees that Book A is in the "to be revised" state with his name next to it. He goes about the task of revising the document, which takes some number of weeks.

5. At the same time Dawn sees her assignment and starts working on Book B.

6. Edgar and Francis both get notification emails telling them that they are responsible for reviewing and approving Book A and Book B and to coordinate with Carl and Dawn to schedule the reviews.

7. Dawn finishes her writing work and does all the validation checks required. Using the workflow management interface, she requests promotion of Book B from the "to be revised" state to the "under review" state. All the preconditions are met, so the promotion succeeds.

8. Edgar and Francis both get notifications from the workflow system telling them that Book B is now under review and they should therefore review it and approve or reject it. They review the document, have a meeting with Dawn, and decide that the document needs more work. Using the workflow management interface, they both indicate rejection of the document. Dawn is sent a notification that the document was not approved as a result of the review.

9. Dawn promotes Book B to the only available state, which is "under revision" state. She does more work, then promotes it again to the "under review" state.
10. Again Edgar and Francis review the document and this time approve it. Dawn is sent a notification that her document has been approved.
11. Dawn promotes Book B to the "approved" state. A PDF and HTML rendition of the book are automatically generated and this version of Book B and its revisions are added to the "docs for internal use" folder in the project's area on the corporate Intranet.
12. Dawn attempts to promote Book B to the "published" state but is told that she does not have the authority to promote objects to the "published" state.
13. Dawn asks Barbara to do the promotion. Barbara attempts the promotion and is told that Book A must also be in the "approved" state before either document can be published.
14. Carl goes through the same process with Book A and eventually gets it approved. Carl promotes Book A to the "approved" state.
15. Dawn gets a notification that the preconditions for promotion of Book B have been met. Dawn tells Barbara that Book B can now be published.
16. Barbara attempts to promote Book B to the "published" state and notices that Book A is also promoted at the same time. The previously-generated renditions of the books are moved into the "customer ready" area of the corporate Intranet and a notification is sent to Manufacturing that these books are ready for packaging and shipment.

## System's View of the Scenario

The following describes the system actions that underlying the scenario:

1. The output of Arthur's analysis and development activity includes the following items:
    - A workflow template document
    - A set of activity policy scripts that implement the precondition checking and post condition checking and side effects
    - A set of configurations for the workflow management system that implement the property-related business rules.
    - Definitions of the user roles and the business rules for sending them notifications when states and/or properties change.
2. When Barbara creates the new concrete workflow from the workflow template, the system generates a concrete workflow document and adds it to the system. The system binds actual users to the roles defined by Arthur.

    When Barbara promotes the documents to the "under revision" state, she is actually initiating traversal from each ManagedObject element to the target state (through the workflow management UI, which is implementing the traversal action in this case). The workflow management system communicates with the underlying link management system (which is serving up the hyperdocument constructed from the newly-created concrete workflow document) to indicate that it is attempting traversal from a member of a hyperlink anchor.

The link management system sees that there is an activity policy associated with this action and that it applies to the node in question (the ManagedObject element). This activity policy checks the preconditions for the target (the anchor member of the PotentialState anchor, which should be a State element which then points to a StateTemplate element which then gets you to the preconditions for that state) and, seeing that they have been satisfied, responds with a "true" from the "apply activity policy" request, allowing the traverse-from action to continue.

The workflow engine expresses the "traverse-to" action by sending another message to the link management system telling it that it is attempting to traverse to the State member of the PotentialState anchor. Again the link management system sees that there is an applicable activity policy. This policy checks that the user doing the traversal is allowed to perform the traverse-to action and, seeing that she is, continues by generating the required notifications. The policy then constructs a new version of the concrete workflow document that reflects the movement of the Managed Object from its starting state to its new state. This new version is saved in the underlying storage repository as a new version of the concrete workflow document. Finally, the policy script checks the post conditions for the state to make sure that they are all true (they are). The policy returns from the "apply activity policy" request with a "continue" response and the traverse-to action continues.

The workflow management system sees that there is now a new version of the concrete workflow document and updates the hyperdocument to reflect this new version. The workflow management user interface now reflects the new version of the document, which shows Document A in the "under revision" state instead of the "To be revised" state.

All of the remaining steps are variations on these same sequences of actions.

# Conclusions and Further Research

More implementation experience is required to determine the true value, especially in the long term, for hyperlink-based workflow representation and implementation.

There is more analysis work to be done to abstract out user roles and tasks so that those can be captured in a more generic way, with the goal of reducing the burden on integrators to implement workflow instances.

# Bibliography

**[HyTime]** ISO/IEC 10744:1997 Information Technology -- Hypermedia/Time-Based Structuring Language (HyTime).

**[XLink]** XML Linking Language, W3C

**[13250]** ISO/IEC 13250 Topic Maps: Information Technology -- Document Description and Markup Languages

**[XTM]** XML Topic Maps, Published by Topicmaps.org.

**[WFMC]** The Workflow Reference Model, Doc Number TC00-1003, 19 Jan 95

# Footnotes

1 The XLink specification appears to allow a link to have only one anchor. However, I do not consider this meaningful so am constraining links to two distinct anchors in this paper.

# Biography

**W. Eliot Kimber**
DataChannel, Inc.
Austin
TX
78757
E-mail: eliot@isogen.com
Web: www.isogen.com

W. Eliot Kimber is Lead Brain for DataChannel. Eliot is a founding member of the XML Working Group, Co-editor of ISO/IEC 10744:1997 (HyTime), and Co-Editor of ISO/IEC 10743, Standard Music Description Language. Eliot is also involved in the STEP and SGML Harmonization effort, which led to a deeper appreciation of the power and utility of formal data modeling as a design and analysis tool. Eliot writes and speaks frequently on the subject of SGML, XML, hyperlinking, and related topics. When not trying to wrestle chaotic data into orderly structures, Eliot enjoys swimming and biking and guitar playing. Eliot is a devoted husband and dog owner.